ANNUAL STATUS REPORT

Error Control Techniques for Satellite
and Space Communications
NASA Grant Number NAG5-557

Principal Investigator:
Daniel J. Costello, Jr.

November 1992

Summary of Progress

1) Construction of Robustly Good Trellis Codes for Use With Sequential Decoding

Sequential decoding has been shown to be the best alternative for achieving large coding gains with trellis coded 8-PSK and 16-QAM modulation [1]. Preliminary results reported earlier on code performance were based either on codes designed to maximize asymptotic coding gain, i.e., optimum free distance (OFD) codes, or on codes designed to maximize the computational speed of sequential decoding, i.e., optimum distance profile (ODP) codes. More detailed studies have since shown that the best overall performance is not achieved with either OFD codes or with ODP codes.

Rather, a new approach has been developed for constructing trellis codes which are neither OFD nor ODP. We call the new codes robustly good trellis codes. Given that a robustly good trellis code of constraint length v has been found, the approach used to find a constraint length $v+1$ robustly good trellis code is to find the code that improves the free distance or the distance profile of the constraint length v code, with priority given to improving the free distance. In other words, we try to find a longer code which has a free distance or a distance profile superior to or identical to the shorter one. Systematic feedback 8-PSK and 16-QAM robustly good trellis codes with v up to 15 and asymptotic coding gains up to 6.66 dB are obtained using this approach. Compared to ODP and OFD trellis codes, the robustly good trellis codes provide a much better trade-off between free distance and distance profile. Indeed, the new codes achieve nearly the same free distances as the OFD codes and nearly the same distance profiles as the ODP codes.

A paper based on these new results is being prepared for submission to the IEEE Transactions on Information Theory. A summary of this paper, which will be presented at the 1993 IEEE International Symposium on Information Theory, is included as Appendix A of this report [2].

2) Unequal Error Protection Capabilities of Convolutional Codes

An important practical problem in many coding applications is unequal error protection (UEP). Although this problem receives little attention in the literature, it can be of great importance in applications such as image transmission from deep space, where different parts of the data stream, representing, say, important feature information rather than background scenes, must be protected with higher reliability than the rest of the data. Algebraic block codes which have UEP capabilities have been studied by some researchers. But little work has been done on the UEP properties of the convolutional codes most often found in space and satellite applications.

We have recently begun a study of the UEP capabilities of (n, k, m) convolutional codes with $k > 1$. The usual transfer function analysis technique is modified to calculate an upper bound on the bit error rate (BER) $P_b^{(i)}(E)$ for each input bit position $i, 1 \leq i \leq k$. We also

define a *distance vector* $\bar{d}$ for convolutional codes as follows:

$$\bar{d} = (d_f^{(1)}, d_f^{(2)}, \cdots, d_f^{(k)}), \tag{1}$$

where $d_f^{(i)}$ is the *effective minimum free distance* seen by input bit position $i, 1 \leq i \leq k$. $\bar{d}$ is then a measure of the UEP properties of a given code, i.e., if $\bar{d}$ is a constant vector, then all input bit positions have equal error protection, but if $d_f^{(i)} > d_f^{(j)}$ for some i and j, then input bit position i has greater error protection than input bit position j. Most optimum free distance (OFD) convolutional codes are found to have constant distance vectors, i.e., they provide no UEP capability. One of the goals of our research is to construct convolutional codes with UEP properties, i.e., for a given desired distance vector, find the encoder realization with the minimum overall memory.

A paper based on the modified transfer function analysis technique of UEP capabilities will be presented at the 1993 IEEE International Symposium on Information Theory. A summary of this paper is included as Appendix B of this report [3]. Additional progress on this problem will be presented in our next report.

3) New Results on Rate $1/n$ Convolutional Codes

The problem of finding good large constraint length, low rate convolutional codes for deep space applications is again being investigated. An intriguing new formula for computing the free distance of rate $1/n$ convolutional codes has been discovered. This formula is based on correlation coefficients of both the information sequence and the generator sequence. It allows us to show that for the class of randomly constructed $(n, 1, m)$ convolutional codes

$$\lim_{m \to \infty} \frac{d_{free}}{n(m+1)} = \frac{1}{2}, \tag{2}$$

with probability 1, i.e., in the limit of large constraint length, the free distance of almost all codes approaches one-half the constraint length. This result is consistent with what is known for short OFD codes, but suggests the existence of much better long codes than was previously thought possible.

The new formula for computing free distance has been used to construct some large constraint length codes with excellent distance properties. These codes are close to optimal for small constraint lengths and their distances continue to grow as suggested by (2) for larger constraint lengths. The new codes are extremely powerful and would make good candidates for use with sequential decoding in deep space applications.

A paper based on this new formula will be presented at the 1993 IEEE International Symposium on Information Theory. A summary of this paper is included as Appendix C of this report [4]. Further progress on this problem will be presented in our next report.

4) New Results on Double Memory Convolutional Codes

Unit Memory (UM) convolutional codes, i.e., $(n, k, 1)$ codes with one memory unit per input bit position, were introduced by Lee [5]. UM codes were found to have good distance properties and their byte orientation (k bits per byte) made them desirable as inner codes in concatenated systems with symbol based (Reed Solomon) outer codes. We have begun to study Double Memory (DM) codes, i.e., $(n, k, 2)$ codes with two memory units per input bit position. We feel that DM codes may have even better distance properties than UM codes, while maintaining their byte orientation ($2k$ bits per byte). This belief is based on a new upper bound which indicates that the free distance of DM codes can be larger than the free distance attained by other codes with the same rate and encoder memory. We are currently conducting a search for optimal DM codes. These codes will be good candidates for use in concatenation systems.

A paper based on these results was presented at the 1992 Allerton Conference on Communications, Control, and Computing. A copy of this paper is included as Appendix D of this report [6]. The results of our search for optimum DM codes will be presented in our next report.

5) Constructing Convolutional Codes from Quasi-Cyclic Codes

The lack of a suitable algebraic structure has long proved a hindrance to researchers trying to construct good long convolutional codes. Although some connections between cyclic codes and convolutional codes have been found, few good new convolutional codes have been constructed using these connections. Recently, Tanner [7] suggested an interesting link between quasi-cyclic codes and convolutional codes. We have extended Tanner's work and developed an algorithm for constructing a convolutional code from a given quasi-cyclic code. The free distance of the constructed convolutional code is lower bounded by the minimum distance of the quasi-cyclic code. Some long convolutional codes with large (but suboptimal) free distances have been found using this construction. Our hope is that additional development of this theory will yield large classes of good long convolutional codes.

A paper based on these results was presented at the 1992 DIMACS Conference on Coding and Quantization. A copy of this paper is included as Appendix E of this report [8]. Further results will be presented in our next report.

References

[1] D. J. Costello, Jr., L. C. Perez, and F. Q. Wang, "Bandwidth Efficient CCSDS Coding Standard Proposals", Semi-Annual Status Report, NASA Grant NAG5-557, May 1992.

[2] F. Q. Wang and D. J. Costello, Jr., "On the Design Criteria for Trellis Codes with Sequential Decoding", IEEE International Symposium on Information Theory, San Antonio, Texas, January 1993.

[3] D. G. Mills and D. J. Costello, Jr., "Using a Modified Transfer Function to Calculate Unequal Error Protection Capabilities of Convolutional Codes", IEEE International Symposium on Information Theory, San Antonio, Texas, January 1993.

[4] Y. Levy and D. J. Costello, Jr., "A New Bound on the Free Distance of Rate $1/n$ Convolutional Codes", IEEE International Symposium on Information Theory, San Antonio, Texas, January 1993.

[5] L. N. Lee, "Short Unit-Memory Byte-Oriented Binary Convolutional Codes Having Maximal Free Distance", *IEEE Transactions on Information Theory*, IT-22, pp. 349-352, May 1976.

[6] D. G. Mills and D. J. Costello, Jr., "An Upper Bound on the Free Distance of Double Memory Convolutional Codes", Allerton Conference on Communications, Controls, and Computing, Monticello, Illinois, September 1992.

[7] R. M. Tanner, "Convolutional Codes from Quasi-Cyclic Codes: A Link Between the Theories of Block and Convolutional Codes", University of California at Santa Cruz, Computer Research Laboratory Technical Report USC-CRL-87-21, November 1987.

[8] Y. Levy and D. J. Costello, Jr., "An Algebraic Approach to Constructing Convolutional Codes from Quasi-Cyclic Codes", DIMACS Conference on Coding and Quantization, Piscataway, New Jersey, October 1992.

APPENDIX A

On the Design Criteria for Trellis Codes with Sequential Decoding

On the Design Criteria for Trellis Codes with Sequential Decoding [1]

Fu-Quan Wang
Daniel J. Costello,Jr.

Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

Submitted as a short paper to the 1993
IEEE International Symposium on Information Theory

July 1, 1992

Abstract

In this paper, the design criteria for trellis codes with sequential decoding are examined. A comparision of trellis codes with Optimum Distance Profile (ODP) and Optimum Free Distance (OFD) reveals that both ODP and OFD trellis codes for some constraint lengths may not result in the best trade-off between error performance and computational performance when sequential decoding is used. A new approach is proposed to construct robustly good trellis codes for use with sequential decoding. The new codes obtained using this approach achieve nearly the same free distances as the OFD codes and nearly the same distance profiles as the ODP codes.

[1]This work was supported by NSF grant NCR 89-03429 and NASA grant NAG 5-557.

SUMMARY

Most of the trellis codes constructed thus far have been for use with the Viterbi algorithm[1, 2]. The asymptotic error performance of the Viterbi algorithm[3] is determined by the minimum free Euclidean distance of the code. Thus, the free distance has been used as the main criterion in code construction for use with the Viterbi algorithm[1, 2]. However, the computational effort of the Viterbi algorithm grows exponentially with the code constraint length ν. This limits its application to codes with small values of ν and relatively modest coding gains. To achieve larger coding gains with tolerable computational complexity, alternative decoding algorithms must be used.

It is well known that sequential decoding [4]-[6] can perform almost as well as the Viterbi algorithm and its computational complexity is essentially independent of ν. Thus, larger coding gains are possible when larger constraint length codes are used with sequential decoding. In [7, 8, 9], sequential decoding has been used to decode trellis codes and these papers demonstrate that sequential decoding is a good alternative to the Viterbi algorithm. However, very few papers have addressed the problem of constructing trellis codes for use with sequential decoding. In this paper, trellis codes with Optimum Distance Profile (ODP) and Optimum Free Distance (OFD) are examined and design criteria for trellis codes with sequential decoding are discussed. We show that neither the ODP nor the OFD trellis codes provide the best trade-off between distance profile and free distance. Thus, a new algorithm is proposed to construct robustly good trellis codes.

First, we show that the computational distribution of sequential decoding for trellis codes is a function of the code's column distance function. Consider a rate $k/k+1$ trellis codes. For a partial path associated with a message m of length l branches ($k \times l$ information bits), the cumulative Fano metric is given by

$$L(l) = -\sum_{i=0}^{l} \{\alpha d^2[z_i, a_i^m] + \beta(z_i)\}, \tag{1}$$

where $d^2[z_i, a_i^m] = ||z_i - a_i^m||^2$, z_i is the received signal, a_i^m is the hypothetical transmitted signal, α is a positive constant, and $\beta(z_i)$ is a constant independent of the

transmitted signal. It can further be shown that $L(l)$ is upper bounded by

$$L(l) \leq -\alpha d_l^2 + \sum_{i=0}^{l} \beta(z_i), \tag{2}$$

where d_l^2 is the column distance function of the code[10]. A sequential decoder abandons a path whenever the Fano metric falls below the metric of a temporarily more likely path. From (1) it follows that a partial path has a small path metric and is rejected by the decoder if its distance from the revceived sequence is sufficiently large. But it is the speed of this rejection that deterimines the computational effort. Without loss of generality, we assume that the decoder follows a wrong path from the original node. Then, we have the upper bound of the path metric given by (2). This bound shows that the metric function along any path different from the correct path decreases at least as fast as the column distance function grows. Thus, fast rejection of an incorrect path requires a rapidly decreasing metric along incorrect paths. Consequently, a rapidly increasing column distance function guarantees fast decoding. This observation has long been recognized for convolutional codes [10]. From the above analysis, we see that a similar conclusion can be drawn for trellis codes. We give an example to verify this. The column distance functions (CDF's) of two $\nu = 9$ 8-PSK trellis codes are shown in Figure 1. Code 1 has parity-check coefficients $H^0 = 1761$, $H^1 = 0106$, and $H^2 = 0100$ in octal form. The parity-check coefficients for code 2 are $H^0 = 1001$, $H^1 = 0036$, and $H^2 = 0546$. Both codes have the same free distance $d_{free}^2 = 6.343$. However, note that the CDF of code 1 grows much faster than code 2. Figure 2 shows the computational distributions of the two codes at an $SNR = 7.7$ dB. It is seen that the computational behavior of code 1 is superior to code 2. This example shows that a rapidly increasing column function results in good computational performance. This is consistant with the results for convolutional codes[10]. It can also be shown that the initial part of the CDF (called the distance profile) plays a more important role than the latter part. Thus, the distance profile should be optimized to achieve good computational performance.

A trellis code is said to have a distance profile $(d_0^2, d_1^2, \cdots, d_\nu^2)$ superior to the distance profile $(d_0'^2, d_1'^2, \cdots, d_\nu'^2)$ of another code of the same constraint length ν if for

some p, $0 \leq p \leq \nu$,

$$\begin{aligned} d_i^2 &= d_i'^2, \quad i = 0, 1, \cdots, p-1 \\ &> d_i'^2, \quad i = p. \end{aligned} \tag{3}$$

We say a code is an optimum distance profile code if its distance profile is equal to or superior to that of any other code with the same constraint length. Trellis codes with optimum distance profiles can be constructed by computer search. In the construction algorithm, the free distance should be used as a secondary criterion, i.e., the code having the larger free distance is retained whenever two codes have the same distance profile. Compared with the Ungerboeck codes, we found that the ODP trellis codes have much smaller free distances for some constraint lengths. For example, the free distance of ODP trellis coded 8-PSK with $\nu = 7$ is only 4.0 compared with 6.59 for the Ungerboeck code. This results in a reduction of more than 2.0 dB in asymptotic coding gain. Thus, it appears that ODP codes do not provide a good trade-off between free distance and distance profile.

We have also conducted exhaustive searches for OFD trellis codes in which the distance profile was used as a secondary criterion. Our results indicate that the OFD trellis codes do not provide the best trade-off between distance profile and free distance, either. Figure 3 shows the distance profiles of ODP, OFD, and Ungerboeck (UG) trellis coded 8-PSK with $\nu = 7$. Note that the OFD code has a much inferior distance profile than the ODP code. (Ungerboeck did not use the distance profile as a secondary criterion in his code construction [1, 2]. Thus, the UG code has an inferior distance profile compared to the OFD code, although both codes have the same free distance.)

Thus, we have constructed trellis codes which are neither optimum free distance nor optimum distance profile. We call the new codes Robustly Good Codes (RGC). Given that a robustly good trellis code of constraint length ν has been found, the approach used to find a constraint length $\nu+1$ robustly good trellis code is to find the code that improves the free distance or the distance profile of the constraint length ν code, with priority given to improving the free distance. In other words, we try to find a longer code which has a free distance or a distance profile superior to or

identical to the shorter one.

Suppose that the free distance and distance profile of a robustly good trellis code with constraint length ν are $d^2_{free}(\nu)$ and $d^2(\nu) = \{d^2_0(\nu), d^2_1(\nu), \cdots, d^2_\nu(\nu)\}$, respectively. Then a robustly good trellis code with constraint length $\nu + 1$ can be found using the following algorithm:

0) Set $d^{2\prime}_{free} = d^2_{free}(\nu)$ and $d^{2\prime} = \{d^{2\prime}_0, d^{2\prime}_0, \cdots, d^{2\prime}_\nu, d^{2\prime}_{\nu+1}\} = \{d^2_0(\nu), d^2_1(\nu), \cdots, d^2_\nu(\nu), d^2_\nu(\nu)\}$.

1) Select a new code C by systematically changing the parity-check coefficients. Set $i = 0$.

2) Compute the column distance d^2_i of code C.

3) If $d^2_i < d^{2\prime}_i$, go to 8). Otherwise $i \leftarrow i + 1$, go to 4).

4) If $i \leq \nu + 1$, go to 2). Otherwise, go to 5).

5) Compute the free distance d^2_{free} of code C. If $d^2_{free} > d^{2\prime}_{free}$, print the parity-check coefficients of code C, d^2_{free}, $d^2 = \{d^2_0, d^2_1, \cdots, d^2_{\nu+1}\}$, and "a better free distance code is found". Otherwise, go to 6).

6) If $d^2_{free} < d^{2\prime}_{free}$, go to 8). Otherwise, go to 7).

7) If $d^2_i > d^{2\prime}_i$ for some i, print the parity-check coefficients of code C, d^2_{free}, d^2, and "a better distance profile code is found".

8) If the set of codes is exhausted, stop. Otherwise, go to 1).

The above algorithm guarantees finding a trellis code that is no worse than the previous constraint length code in terms of free distance and distance profile. The initial code can be chosen such that it results in a good trade-off between distance profile and free distance. We began our construction of robustly good trellis codes at a constraint length of 3. Trellis codes for 8-PSK modulation constructed using this approach are shown in Table I where d^2_ν is the minimum distance and d^2_{free} is the free distance. (d^2_ν is a good indicator of the distance profile of a code.) The minimum distances and free distances of Ungerboeck (UG) and Porath and Aulin ($P\&A$)[11] codes were also included for comparision. Compared to ODP and OFD trellis codes, the robustly good trellis codes provide a much better trade-off between free distance and distance profile. Indeed, the new codes achieve nearly the same free distances as the OFD codes and nearly the same distance profiles as the ODP codes. Trellis codes for 16-QAM modulation have also been constructed using this approach.

References

[1] G. Ungerboeck, "Channel coding with multilevel/phase signals", *IEEE Trans. Inform. Theory*, **IT–28**, pp. 55–67, January 1982.

[2] G. Ungerboeck, "Trellis Coded Modulation with Redundant Signal Sets, Part II: State of the Art", *IEEE Commun. Mag.*, **Vol. 25**, pp. 12–22, February 1987.

[3] A. J. Viterbi, " Error Bounds for Convolutional Codes and an Asymptotically optimum Decoding Algorithm," *IEEE Trans. Inform. Theory*, **IT–13**, pp. 260–269, April 1967.

[4] K. Zigangirov, " Some Sequential Decoding Procedures," *Probl. Peredachi Informatsii.*, **Vol. 2**, pp. 13–25, 1966.

[5] F. Jelinek, " A Fast Sequential Decoding Algorithm Using a Stack," *IBM J. Res. Develop.*, **Vol. 13**, pp. 675–685, November 1969.

[6] R. M. Fano, " A Heuristic Discussion of Probabilistic Decoding," *IEEE Trans. Inform. Theory*, **IT–9**, pp. 64–74, April 1963.

[7] G. J. Pottie and D. P. Taylor, " A Comparison of Reduced Complexity Decoding Algorithms for Trellis Codes," *IEEE J. Sel. Areas Commun.*, **SAC–7**, pp. 1369–1380, December 1989.

[8] F. Q. Wang and D. J. Costello,Jr., "Erasurefree Sequential Decoding and Its Application to Trellis Codes", presented at the 1990 IEEE Int. Symp. Inform. Theory, San Diego, CA, January 1990.

[9] F. Q. Wang and D. J. Costello,Jr., "A Hybrid M-algorithm/Sequential Decoder for Convolutional and Trellis Codes", in *Proc. 1990 Int. Symp. Inform. Theory and Its Applications*, Hawaii, pp. 67–69, November 1990.

[10] P. R. Chevillat and D. J. Costello,Jr., "Distance and Computation in Sequential Decoding," *IEEE Trans. Commun.*, **COM–24**, pp. 440–447, April 1976.

[11] J. Porath and T. Aulin, " Algorithmic Construction of Trellis Codes," submitted to the *IEEE Trans. Commun.*, November 1990.

Table I. Robustly good trellis codes for 8-PSK modulation

ν	H^0	H^1	H^2	d^2_ν			d^2_{free}		
				RGC	UG	P&A	RGC	UG	P&A
3	15	06	04	2.59	2.59	–	4.59	4.59	–
4	35	12	10	3.17	2.59	2.59	5.17	5.17	5.17
5	67	26	20	3.17	3.17	3.17	5.17	5.76	5.76
6	121	066	060	3.76	2.59	2.59	6.00	6.34	6.34
7	337	026	100	3.76	2.59	2.59	6.34	6.59	6.59
8	701	166	300	4.34	2.59	2.59	6.93	7.52	7.52
9	1175	0142	0400	4.34	3.76	3.17	6.93	7.52	7.52
10	2015	0402	0400	4.34	3.17	3.17	7.76	7.52	8.10
11	4047	2302	0400	4.93	–	3.17	8.10	–	8.34
12	10517	06462	04400	4.93	–	3.76	8.34	–	8.69
13	33001	16266	01400	4.93	–	3.76	8.69	–	8.69
14	57001	22266	35400	5.52	–	–	8.69	–	–
15	104001	045666	035400	5.52	–	4.34	9.27	–	9.51

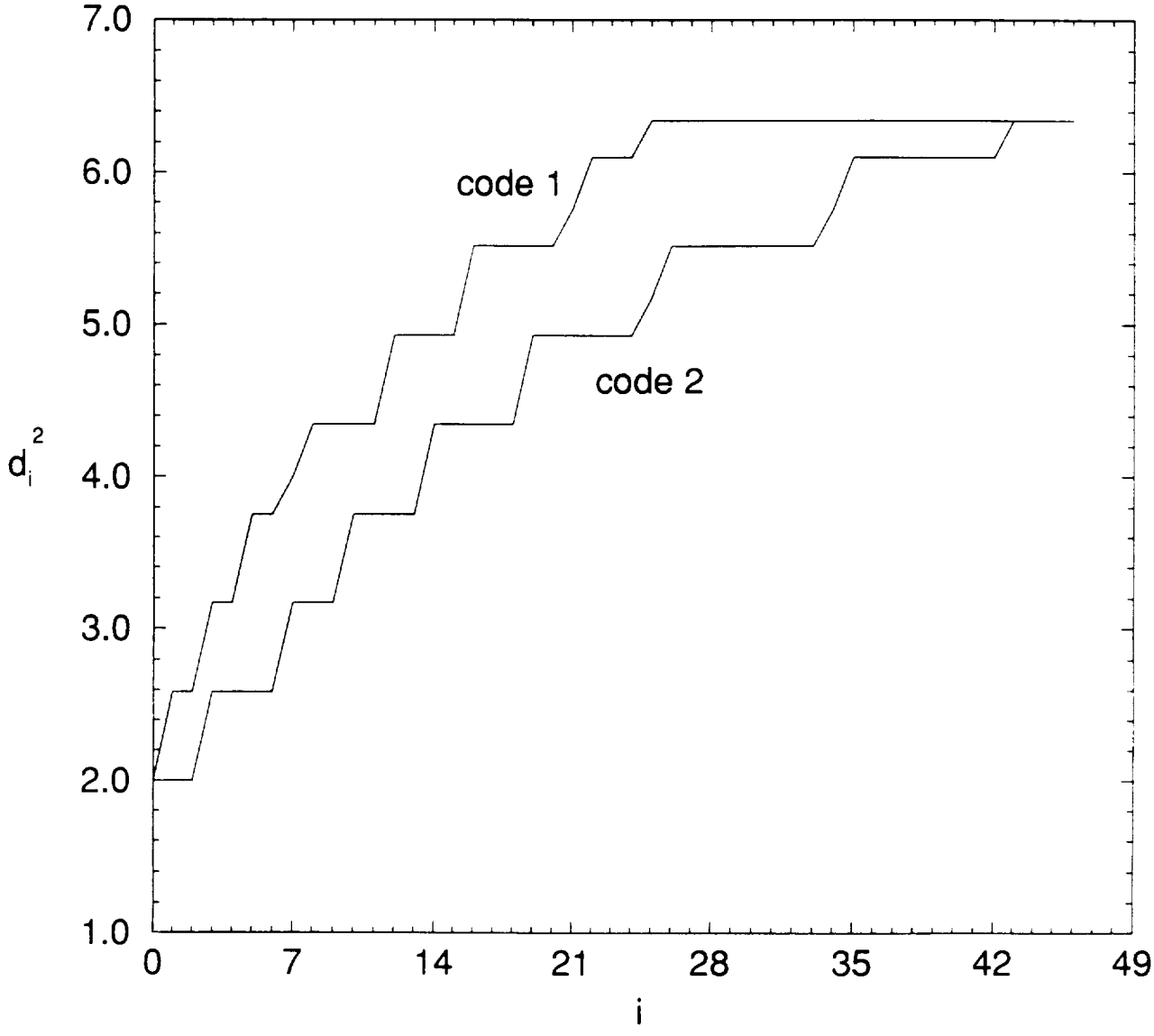

Figure 1. CDF's of two ν=9 trellis codes

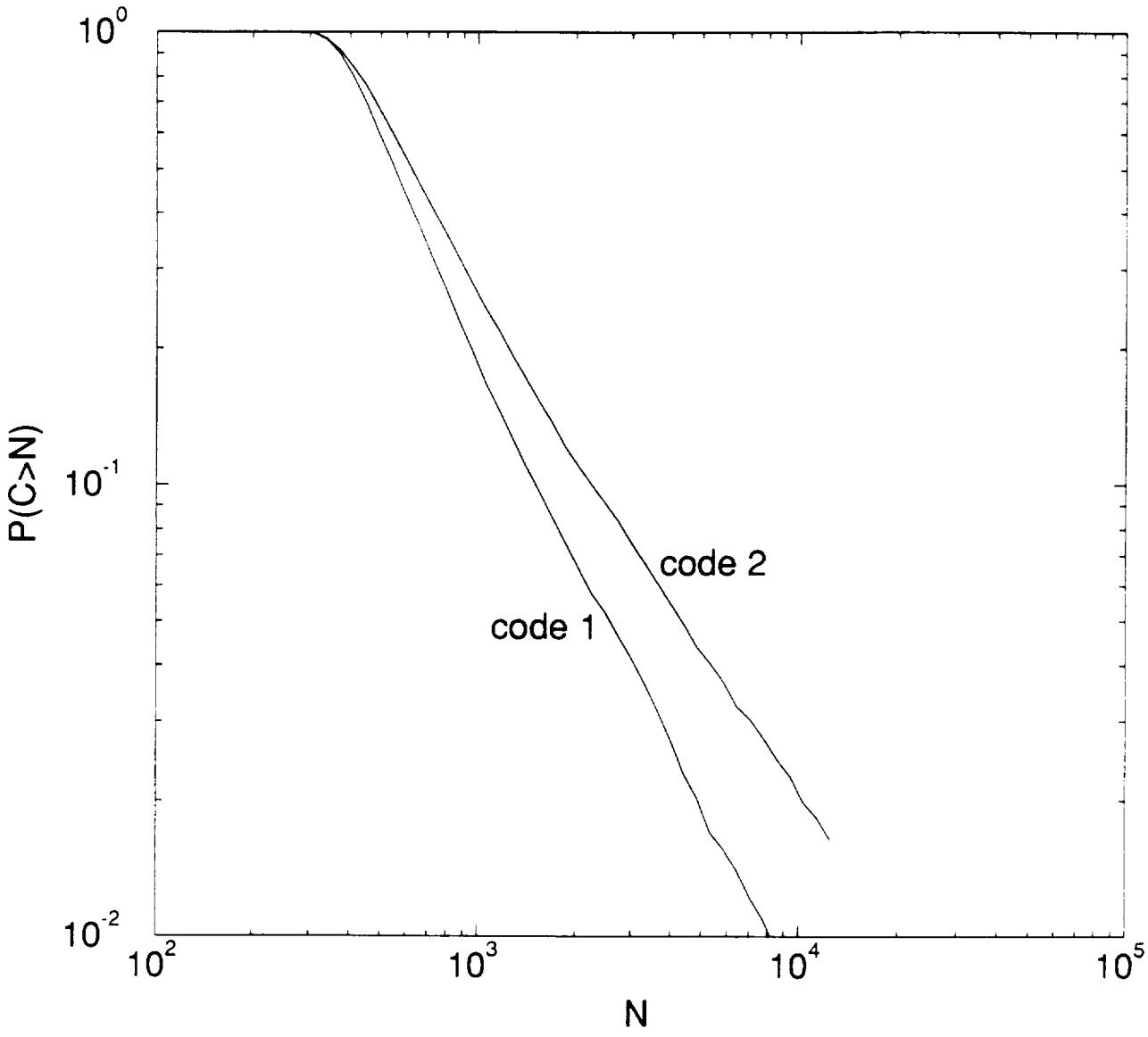

Figure 2. Computational distribitions of tw0 ν=9 trellis codes

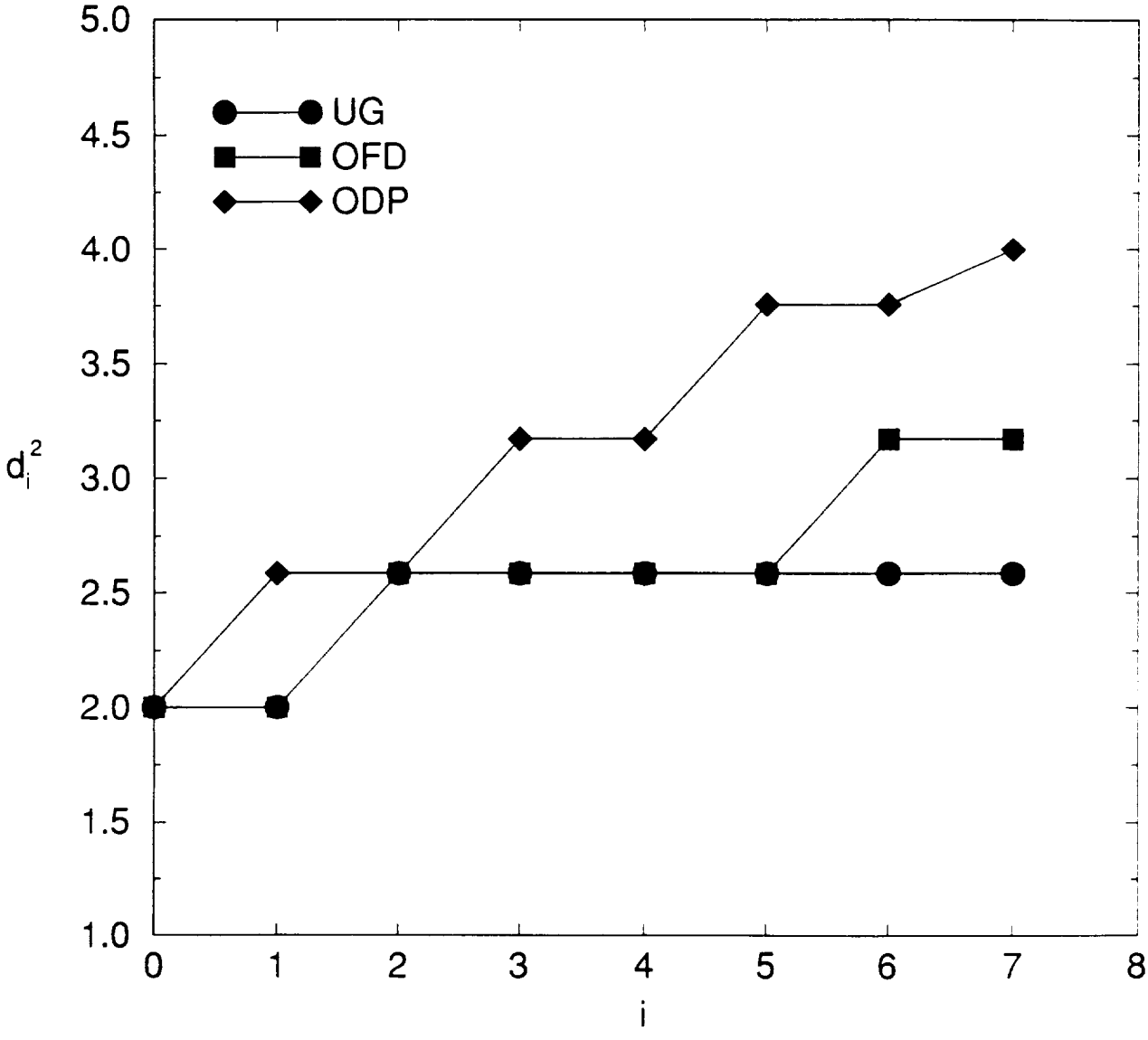

Figure 3. Distance profile comparision of three ν=7 trellis codes

APPENDIX B

Using a Modified Transfer Function to Calculate Unequal Error Protection Capabilities of Convolutional Codes

Using a Modified Transfer Function to Calculate Unequal Error Protection Capabilities of Convolutional Codes [1]

Diane G. Mills
Daniel J. Costello, Jr.

Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

Submitted as a short paper to the 1993
IEEE International Symposium on Information Theory
July 1, 1992

Abstract

This paper proposes a modified transfer function analysis that yields the individual bit error probability for any specified input bit position. This analysis proves useful in determining the unequal error protection (UEP) capabilities of convolutional codes. The UEP transfer function is used to determine an upper bound on the bit error probabilities for individual input bit positions in convolutional codes. The form of the individual bit error probability bound reveals three factors that affect the UEP capabilities of a convolutional encoder: the effective free distance of each bit position, the number of low weight code vectors, and the distribution of 1's in the input sequences that generate low-weight code vectors.

---

[1] This work was supported by NSF grant EID 90-17558, NSF grant NCR 89-03429, and NASA grant NAG5-557.

Summary

State diagram analysis has long been used to determine the transfer functions of low complexity (n,k,m) convolutional encoders. The transfer function, in turn, is then manipulated to determine the free distance, event error probability, and bit error probability of the encoder. The bit error probability derived from the standard transfer function is the probability that an input bit is decoded incorrectly. However, the error probability which is relevant for unequal error protection (UEP) codes is the probability of bit error at each specific input position. UEP codes are of interest in several environments. For instance, in packet switched networks, the header information requires more error protection than the data. UEP codes could provide the additional header protection. In a multi-user environment, different users may require more or less error protection than others. UEP codes may again be appropriate.

This paper proposes a modified transfer function analysis that yields the individual bit error probability for any specified input bit position. The method is described for codes with $k>1$, but can easily be applied to codes with $k=1$ by first transforming the code to a unit memory coder or double memory code [1,2]. First, standard transfer function analysis will be reviewed. Next, the modified transfer function will be described and illustrated with an example. An algorithm that calculates the modified transfer function is briefly described. A discussion of results then follows.

An *(n,k,m)* convolutional encoder accepts k-dimensional input vectors, adds redundancy according to the encoding rule, and then outputs n-dimensional code vectors. The parameter m is the maximum number of memory registers needed to store any element of the input vector. The total encoder memory, K, is defined as the total number of memory registers in the encoder.

It is assumed that the reader is familiar with the method of determining a transfer function from an augmented state diagram [3]. The two-variable transfer function is of the form $T(X,Y) = \sum_{\substack{b=1 \\ d=dfree}} A_{b,d} X^d Y^b$. The average bit error probability for a specific transfer function is bounded by $P_b(E) < \frac{1}{k} \sum_d B_d P_d$, where $B_d = \sum_b b A_{d,b}$ is the total number of nonzero information bits associated with all codewords of weight d, and $P_d = 2^d \left[p(1-p)\right]^{\frac{d}{2}}$. (For simplicity, we assume a binary symmetric channel with crossover probability p.)

When the individual bit error probability is desired for each of the k input positions, then the state diagram must be modified before Mason's gain formula is applied. Each branch label becomes $X^i Y_1^{j_1} Y_2^{j_2} \cdots Y_k^{j_k}$, where j_k is equal to the input bit in the k^{th} position, and i is the Hamming weight of the branch output. Obviously, the sum of the j_k 's is the Hamming weight of the input vector. Mason's gain formula is then applied. The resulting UEP transfer function has the form $T(X,Y_1,\cdots,Y_k) = \sum_{d=dfree}^{\infty} \sum_{j=0}^{j_d} C_{d,j} X^d Y_1^{b_{1,j}} \cdots Y_k^{b_{k,j}}$, where $C_{d,j}$ is the number of paths associated with the j^{th} input sequence distribution of 1's that generates code vectors of weight d., j_d is the number of

distinct input sequence distributions that generate code vectors of weight $d.$, and $b_{1,j},\cdots,b_{k,j}$ represents a particular input sequence distribution of 1's. The bound for the individual bit error probability is then $P_b^{(i)}(E) < \sum_d B_d^{(i)} P_d$, $1 \le i \le k$, where $P_b^{(i)}(E)$ is the probability that a bit located in the i^{th} position of the input vector is decoded incorrectly and $B_d^{(i)} = \sum_{j=0}^{j_d} b_{i,j} C_{d,j}$ is the total number of 1's in bit position i of all input vectors that generate code vectors of weight d. Note that the new parameters are related to the original parameters by the equations $B_d = \sum_i B_d^{(i)}$ and $P_b(E) = \frac{1}{k} \sum_i P_b^{(i)}$.

The modified state diagram for a particular (3,2,1) code is shown in Figure 1. The generator vectors for the code are listed in Table 1. The UEP transfer function is

$$T(X,Y_1,Y_2) = X^3\left(Y_2 + Y_1Y_2^2\right) + X^4\left(2Y_1Y_2 + Y_1Y_2^3 + Y_1^2Y_2^2 + Y_1^2Y_2^4\right) + X^5\left(Y_1 + 2Y_1^2Y_2 + 4Y_1^2Y_2^3 + Y_1^3Y_2^2 + 4Y_1^3Y_2^4 + Y_1^2Y_2^5 + Y_1^3Y_2^6\right) + \cdots$$

The bound for the probability of a bit error in the first input position is then $P_b^{(1)}(E) < P_3 + 7P_4 + 33P_5 + \cdots$. Similarly, the bound for the probability of a bit error in the second input position is given by $P_b^{(2)}(E) < 3P_3 + 10P_4 + 43P_5 + \cdots$.

An obvious drawback of state diagram analysis is the high level of complexity when the memory order and input vector dimension are not severely restricted. As the total memory K increases, the number of states increases exponentially. In addition,

as k, the dimension of the input vector increases, the number of branches leaving each state increases exponentially. The number of forward paths, loops, and sets of nontouching loops quickly becomes too unwieldy for analysis. Because of this complexity, an algorithm was developed to calculate the modified transfer function and individual bit error bounds. The algorithm is based on the work in [4], which originally computed the column distance function and transfer function of $(n,1,m)$ convolutional codes. The algorithm was generalized to accept (n,k,m) codes , and modified to compute the newly introduced parameters $B_d^{(i)}$ and $P_b^{(i)}(E)$. The algorithm uses the distance profile of a code to eliminate unproductive paths in the search for the column distance function. It will be a useful tool in developing unequal error protection codes.

Results for a number of existing codes are presented in Table 1. The UEP transfer function verified the expectation that the bit positions with lower memory order generally have a greater bit error probability. However, uneven memory distribution is not required for unequal error protection, which is demonstrated by the individual bit error probabilities of the first and third codes listed in Table 1. While uneven memory distribution is not required, it is expected that as the distribution becomes more uneven, the unequal error protection becomes more pronounced.

Examining the form of $P_b^{(i)}(E)$ and the UEP transfer functions in Table 1, it can be seen that several factors affect the bit error probability for a specific input position. For all of the codes studied

so far, the first term of each $P_b^{(i)}(E)$ has the form $B_{dfree}^{(i)} P_{dfree}$. Note that P_{dfree} is the dominant term of the product. A code for which the first term of each $P_b^{(i)}(E)$ has the form $B_{deff(i)}^{(i)} P_{deff(i)}$ will have significantly more pronounced unequal error protection. The effective free distance for input bit position i , $deff(i)$, is the lowest Hamming weight among all code vectors that are generated by input sequences with at least one 1 in the i^{th} position. The effective free distances are lower bounded by the overall free distance, *dfree*. In addition to the individual effective free distances, two other important factors affecting $P_b^{(i)}(E)$ are the number of low weight code vectors, and the number of 1's in position i that belong to input vectors corresponding to the low weight code vectors. That is, in addition to the traditionally important codeword Hamming weight and multiplicity, the distribution of 1's in the input vector is important. The number of ones in a particular position is related to the length of the input sequence and to the entire Hamming weight of that sequence, although the exact relationship has not been completely determined.

The individual bit error probability bound in this paper has three major features. First, it allows existing codes to be evaluated for unequal error protection. Second, it has revealed a new criterion that must be considered when designing unequal error protection codes, i.e., the distribution of 1's in the input vectors. Last, it should again be noted that the input bit positions of the analyzed existing codes all have an effective free distance equal to the overall free distance. That is, differences in individual bit error protection have

been due only to differences in the multiplicities and input distributions. Using the insights gained from the new UEP analysis technique, we expect to design new codes with different individual effective free distances.

References

1. G.S. Lauer, "Some Optimal Partial-Unit-Memory Codes," *IEEE Transactions on Information Theory, Vol IT-25* , pp. 240-242, March 1979.

2. D.G. Mills and D.J. Costello, Jr., "Description of Double Memory Codes", submitted to 1993 Allerton Conference on Communications, Controls, and Computing.

3. S. Lin and D.J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., 1983.

4. M. Cedervall and R Johannesson, "A Fast Algorithm for Computing Distance Spectrum of Convolutional Codes," *IEEE Transactions on Information Theory, Vol IT-35,* pp. 1145-1159, November 1989.

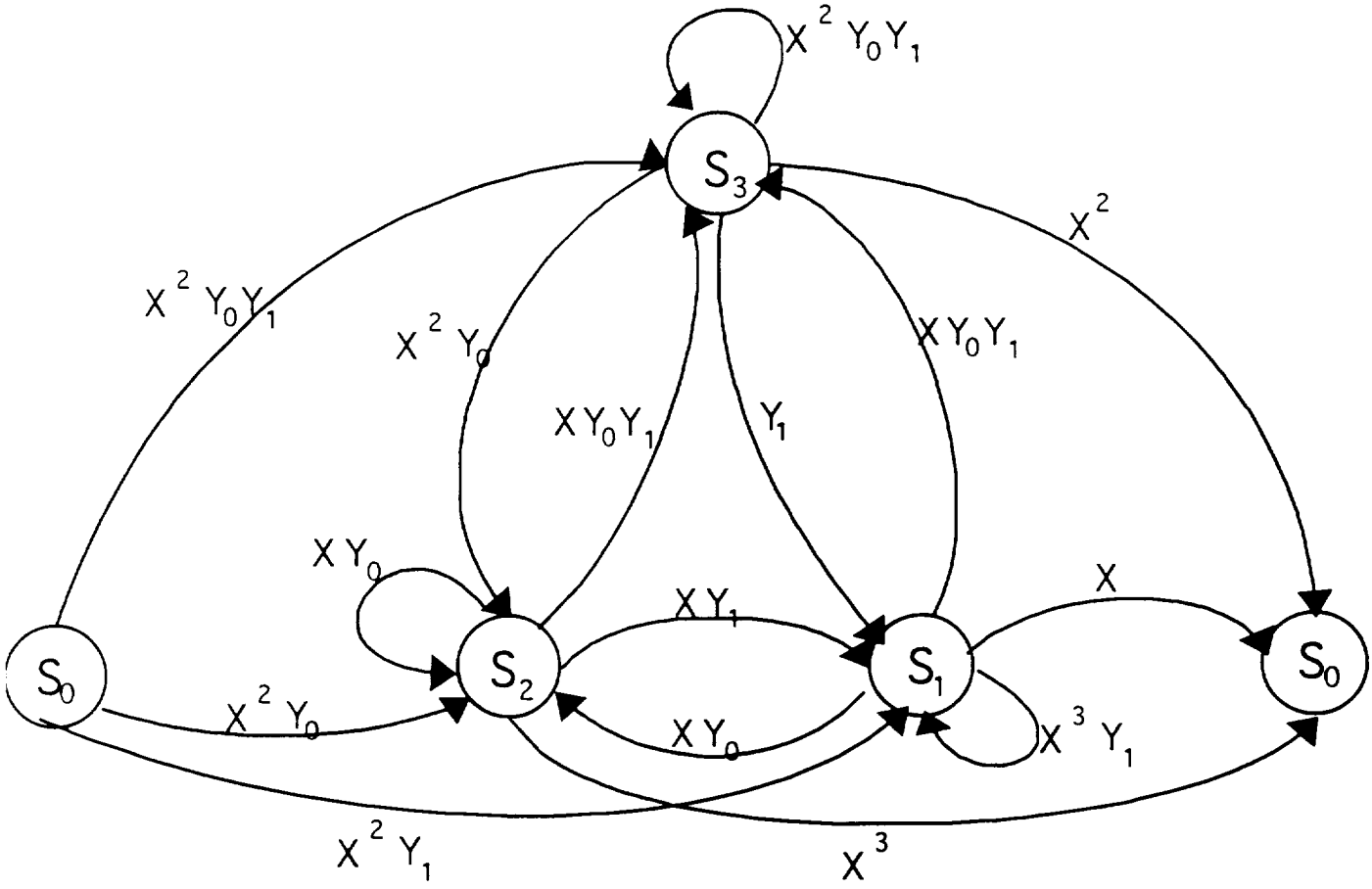
$X^2 Y_0 Y_1$
S_3
X^2
$X^2 Y_0 Y_1$
$X^2 Y_0$
$X Y_0 Y_1$
$X Y_0 Y_1$
Y_1
$X Y_0$
X
$X Y_1$
S_0
S_2
S_1
S_0
$X^2 Y_0$
$X Y_0$
$X^3 Y_1$
$X^2 Y_1$
X^3

Figure 1

Table 1

(n,k,m)	K	Generator Vectors (octal)	UEP Transfer Function	$P_b^{(i)}$
(3,2,1) [3]	2	$g_0^{(0)} = 3 \quad g_0^{(1)} = 1 \quad g_0^{(2)} = 3$ $g_1^{(0)} = 1 \quad g_1^{(1)} = 2 \quad g_1^{(2)} = 2$	$X^3\left(Y_1 + Y_0Y_1^2\right)$ $+X^4\left(Y_1^2 + Y_0^2Y_1^2 + Y_0^3Y_1^3 + Y_0Y_1^3 + Y_0Y_1\right)$ $+\cdots$	$P_b^{(0)} \le P_3 + 7P_4$ $P_b^{(1)} \le 3P_3 + 11P_4$
(3,2,3) [3]	5	$g_0^{(0)} = 14 \quad g_0^{(1)} = 06 \quad g_0^{(2)} = 16$ $g_1^{(0)} = 03 \quad g_1^{(1)} = 10 \quad g_1^{(2)} = 17$	$X^6\left(2Y_0Y_1 + Y_0^2\right)$ $+X^7\left(Y_1 + Y_0^4Y_1^3 + Y_0^2Y_1^3\right)$ $+\cdots$	$P_b^{(0)} \le 4P_6 + 6P_8$ $P_b^{(1)} \le 2P_6 + 7P_8$
(3,2,3) [3]	6	$g_0^{(0)} = 15 \quad g_0^{(1)} = 06 \quad g_0^{(2)} = 15$ $g_1^{(0)} = 06 \quad g_1^{(1)} = 15 \quad g_1^{(2)} = 17$	$X^7(Y_1^2 + Y_0^2Y_1^3 + Y_1^3 + Y_0^4Y_1^8 + Y_0^2Y_1^7$ $+Y_0^2Y_1^5 + Y_0^2Y_1^6 + 2Y_0^2Y_1^4 + Y_1^4 + 2Y_0^2Y_1$ $+Y_0^2 + Y_0^3Y_1^2 + Y_0^3Y_1^3 + Y_0Y_1^2) + \cdots$	$P_b^{(0)} \le 29P_7$ $P_b^{(1)} \le 55P_6$
(6,2,1) [1]	1	$g_0^{(0)} = 2 \quad g_0^{(1)} = 2 \quad g_0^{(2)} = 2$ $g_1^{(0)} = 2 \quad g_1^{(1)} = 3 \quad g_1^{(2)} = 3$ $g_0^{(3)} = 2 \quad g_0^{(4)} = 2 \quad g_0^{(5)} = 2$ $g_1^{(3)} = 2 \quad g_1^{(4)} = 3 \quad g_1^{(5)} = 3$	$X^6\left(Y_1 + 2Y_0Y_1 + Y_0 + Y_0^2Y_1\right)$ $+X^7\left(Y_0Y_1 + 2Y_0^2Y_1 + Y_0^3Y_1\right) + \cdots$	$P_b^{(0)} \le 5P_6 + 8P_8$ $P_b^{(1)} \le 4P_6 + 4P_8$

APPENDIX C

A New Bound on the Free Distance of Rate 1/n Convolutional Codes

A New Bound on the Free Distance of Rate 1/n Convolutional Codes [1]

Y. Levy D.J. Costello, Jr.
Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

19 June 1992

Abstract — A new formula is derived to compute the free distance of rate 1/n convolutional codes. This formula allows us to derive a new asymptotic lower bound on the free distance that surprisingly reaches the upper bound in the limit of large memory m. First, we derive a formula for the weight of the product of two binary polynomials. Then, by considering the free distance as the minimum weight codeword in a convolutional code, we prove that, in the limit of large m, the free distance of rate 1/n convolutional codes is lower bounded by $n(m+1)/2$, i.e., the asymptotic lower bound equals the asymptotic upper-bound. This formula also leads to a new approach for constructing finite constraint length convolutional codes.

Keywords — Convolutional Codes, Lower Bound, Correlation.

[1]This work was supported by NSF Grant NCR89-03429 and by NASA Grant NAG5-557.

1. Introduction

The free distance, d_{free}, is an important parameter in evaluating the performance of convolutional codes. Although it is possible to find optimal convolutional codes with small constraint length by exhaustive search, the free distances of these codes do not indicate what we might expect of codes with larger constraint lengths. Therefore, many researchers have tried to derive lower and upper bounds on the free distance of convolutional codes. Costello [1], Zigangirov and Massey [2] have derived lower and upper bounds on the free distance of fixed and time-varying convolutional codes using a Gilbert-type [3] argument, that is, by deriving bounds on ensemble averages. In this paper, we investigate a new approach to deriving a lower bound on the free distance of rate 1/n convolutional codes. The free distance can be defined as the lowest weight codeword in a linear convolutional code. It is shown that an expression for the weight of the codewords of rate 1/n convolutional codes can be simplified as the constraint length goes to infinity and that the free distance of randomly constructed codes can be computed. Their free distance thus represents a lower bound on the free distance that convolutional codes of these rates can achieve. It is noted that this lower bound meets the asymptotic upper bound on free distance for rate $1/n$ convolutional codes. Although this bound is valid only asymptotically, it suggests that a similar bound might also be found for finite constraint lengths, and it leads to a new approach for constructing finite constraint length convolutional codes.

2. Preliminaries

One of the main problems in deriving bounds on the free distance of binary convolutional codes is the difficulty of computing the weight of the product of two binary polynomials. The goal of this section is to introduce a new way of computing the weight of codewords. The main idea is to convert modulo-2 addition in the binary field into real addition in the integer field. That is, suppose (x, y) are two elements from the binary field $F = \{0, 1\}$, $\oplus$ denotes addition in the binary field, and $+$ denotes addition in the integer field I, then

$$x \oplus y = x + y - 2xy. \tag{1}$$

In order to derive our new formula on the free distance of rate $1/n$ convolutional codes, we need the following definitions:

Definition 1. *Let* $\mathbf{a}(X)$ *be a polynomial with coefficients* a_i. *Then, we define the* 0^{th} *correlation coefficient of* $\mathbf{a}(X)$ *as*

$$R_{0\mathbf{a}} = \sum_{i=0}^{i=\infty} a_i, \tag{2}$$

and more generally the k^{th} correlation coefficients as

$$R_{\mathbf{ka}}(j_1, j_2, \ldots, j_k) = \sum_{i=0}^{i=\infty} a_i a_{i+j_1} \ldots a_{i+j_1+\ldots+j_k}, \tag{4}$$

where $j_1, j_2, \ldots, j_k$ are k integers strictly greater than 0.

Definition 2. *Let $g^{(1)}(X), g^{(2)}(X), \ldots, g^{(n)}(X)$ be the n generator polynomials of a rate 1/n convolutional code C. Then, let $\mathbf{G}(X)$ be the composite generator*

$$\mathbf{G}(X) = g^{(1)}(X^n) + Xg^{(2)}(X^n) + \ldots + X^{n-1}g^{(n)}(X^n), \tag{5}$$

as defined in [4]. Then, for any information sequence $\mathbf{u}(X)$, the code sequence $\mathbf{v}(X)$ is generated by

$$\mathbf{v}(X) = \mathbf{u}(X^n)\mathbf{G}(X). \tag{6}$$

3. Lower bound on the free distance of rate 1/n convolutional codes

Using the previous definitions, we can obtain the following theorem on computing the free distance of rate 1/n convolutional codes.

Theorem 1. *Let C be a rate 1/n convolutional code with composite generator $\mathbf{G}(X)$. Then the free distance of C can be computed as:*

$$d_{free} = \min_{\mathbf{u}(X)\neq 0} \left(R_{0\mathbf{u}} R_{0\mathbf{G}} - 2 \sum_{j=0}^{j=\infty} R_{1\mathbf{u}}(j) R_{1\mathbf{G}}(nj) + 4 \sum_{j,k=0}^{\infty} R_{2\mathbf{u}}(j,k) R_{2\mathbf{G}}(nk, nj) - \ldots \right) \tag{7}$$

(7) gives a general formula for computing the free distance of a rate 1/n convolutional code. However, this formula can be simplified when the constraint length (memory order) goes to infinity. Specifically, let us construct our generator polynomial by randomly selecting its coefficients from F, that is:

$$\mathbf{G}(X) = \sum_{i=0}^{i=n(m+1)-1} g_i X^i, \tag{8}$$

where $g_i \in F = \{0, 1\}$ and $Pr(g_i = 0) = Pr(g_i = 1) = \frac{1}{2}$ for any integer $i > 0$. For these randomly constructed codes, the following theorem can be derived:

Theorem 2. *Let* $\mathbf{G}(X)$ *of degree* $n(m+1)-1$ *be the composite generator of a randomly constructed rate* $1/n$ *convolutional code with memory order* m. *Then, with probability 1,*

$$\lim_{m\to\infty} \frac{d_{free}}{n(m+1)} = \frac{1}{2}. \tag{9}$$

Thus, by taking a random generator $\mathbf{G}(X)$, with probability 1 the free distance is on the order of $n(m+1)/2$ as m goes to infinity. Since there exists a large number of randomly generated codes, and with probability one these codes achieve the free distance of (9), this gives a lower bound on the free distance of rate 1/n convolutional codes, i.e., almost all codes reach this bound. This bound represents a significant improvement on the previous lower bounds derived on the free distance of convolutional codes, since it implies that there exists codes for which the free distance reaches the asymptotic upper-bound derived by Costello [1] and that the number of these codes is very large.

4. Construction of finite constraint length rate 1/n Convolutional Codes.

Although the bound derived in Theorem 2 is only valid as m goes to infinity, Theorem 1 can be used to obtain a lower bound on the free distance of finite constraint length codes, but the bound becomes simple to compute only when m goes to infinity. It is possible, however, that the asymptotic bound is also true for finite m since codes found by exhaustive search for relatively short constraint lengths satisfy this bound. In order to give a better idea of the potential of (7) in computing free distance for finite constraint lengths, we now show that Theorem 1 can lead to a deterministic construction of rate 1/n convolutional codes.

By looking at (7), we note that a code with large free distance requires a large weight generator (large $R_{0\mathbf{G}}$), small first correlation coefficients $R_{1\mathbf{G}}(nj)$, large second correlation coefficients $R_{2\mathbf{G}}(nk, nj)$, etc. Thus, an algorithm can be derived to construct generators of rate 1/n convolutional codes, starting from the all ones generator and replacing ones by zeros as needed to improve the correlation coefficients. A large number of convolutional codes constructed this way have free distances close to optimal codes, and the algorithm allows us to construct codes with much higher constraint lengths than previously constructed codes.

5. Conclusion

A new formula for computing the free distance of rate 1/n convolutional codes is derived. The formula leads to a new asymptotic lower bound on the free distance and to the construction of finite constraint length convolutional codes in a deterministic way. It may also be possible that this formula can be generalized to rate k/n convolutional codes, although the concept of a composite generator does not exist for rates other than 1/n.

References

[1] D.J. Costello, Jr., "Free Distance Bounds for Convolutional Codes," *IEEE Trans. Inf. Theory,* IT-20, No.3, pp.356-365, May 1974.

[2] K.Sh. Zigangirov and J.L. Massey, "Fixed convolutional codes achieve the same bounds as time-varying codes, even at small branch lengths", *Proc. Third. Soviet-Swedish Workshop on Information Theory,* Sochi, May 1987, pp. 335-338.

[3] E.N. Gilbert, "A comparison of signalling alphabets," *Bell Syst. Tech. J.,* vol. 31, pp. 504-522, 1952.

[4] J.L. Massey, D.J. Costello, Jr., and J. Justesen, "Polynomial Weights and Code Constructions," *IEEE Trans. Inf. Theory,* IT-19, pp.101-110, Jan. 1973.

APPENDIX D

An Upper Bound on the Free Distance of Double Memory Convolutional Codes

An Upper Bound on the Free Distance of Double Memory Convolutional Codes [1]

D.G. Mills
D.J. Costello, Jr.

Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

Submitted to the
1992 Allerton Conference on Communications, Controls, and Computing
10 July 1992

Abstract —This paper describes double memory convolutional codes and presents an upper bound on their free distance. It is shown that the free distance upper bound can be larger than the free distances previously attained by codes with the same rates and state complexities.

Keywords — Convolutional Codes, Unequal Error Protection, Upper Bound.

[1]This work was supported by NSF Grant EID90-17558, NSF Grant NCR89-03429, and NASA Grant NAG5-557.

1. Introduction

This paper describes double memory convolutinal codes, which are an extension of the unit memory codes developed by Lee [1]. First, unit memory codes are reviewed, and then double memory codes are presented. Finally, an upper bound on the free distance of double memory codes is developed and examined. Double memory codes are under study as a method of providing unequal error protection [2].

2. Unit Memory Codes

Let u_t and v_t denote the information and code vectors, respectively, at sublock t of a general convolutional code. An (n, k, M) binary convolutional code can be represented by the encoding equation

$$v_t = u_t G_0 + u_{t-1} G_1 + \ldots + u_{t-M} G_M.$$

The information vector for the (n, k, M) code is a k-bit vector, the code vector v_t is an n-bit vector, and the encoding matrices, $G_i, i = 0, 1, \ldots, M$, are kxn binary matrices. The state complexity of a convolutional code is defined to be the number of state variables, $K = Mk$. (For simplicity, it is assumed that the memory is equally allotted to the input bits, i.e., each encoder input is delayed by M memory units.)

A unit memory code (UMC) is a binary convolutional code with memory $M = 1$. The encoding equation of a UMC is $v_t = u_t G_0 + u_{t-1} G_1$. It can be shown that an (n_o, k_o, m) convolutional code with encoding matrices $g_0, g_1, \ldots, g_m$, is equivalent to the $(n = mn_o, k = mk_o, 1)$ UMC which has the encoding matrices

$$G_0 = \begin{bmatrix} g_0 & g_1 & \cdots & g_{m-1} \\ 0 & g_0 & & g_{m-2} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & g_0 \end{bmatrix} \quad G_1 = \begin{bmatrix} g_m & 0 & \cdots & 0 \\ g_{m-1} & g_m & & 0 \\ \vdots & & \ddots & \vdots \\ g_1 & \cdots & & g_m \end{bmatrix}$$

The two codes are equivalent in the sense that the output sequences of the two encoders are identical for identical input sequences [1]. The state complexities of both the (n_o, k_o, m) code and the $(mn_o, mk_o, 1)$ UMC are mk_o.

The free distance, d_{free}, of a convolutional code is the minimum Hamming distance betweeen all pairs of codewords that are associated with input sequences that differ in at least one subblock. It can be assumed without loss of generality that the first

difference between the input sequences appears in subblock 0. Let $v_{t1,t2}$ be the code sequence from time $t1$ to $t2$. For an $(n, k, 1)$ UMC, when the only non-zero portion of the information vector is u_0, then the (non-zero) output is $v_{0,1} = u_0[G_0 G_1]$. The set of all such u_0's and $v_{0,1}$'s forms a $(2n, k)$ block code. It follows that the optimal d_{free} of the $(n, k, 1)$ UMC is upperbounded by the minimum Hamming distance of the optimal $(2n, k)$ block code [1]. The optimal block code minimum distances are tabulated in [3]. In several cases, the UMC upper bound is larger than the free distance attained by the optimal codes with the same state complexity for which the greatest common denominator of n_o and k_o is 1 (hereafter called basic codes). Lee conducted an exhaustive search for the optimal UMCs and found several UMCs better than the optimal basic codes,i.e., the rate and complexity of the optimal UMC and basic codes were identical, but the free distance of the optimal UMC was higher.

3. Double Memory Codes

A double memory code (DMC) is a convolutional code with $M = 2$ that can be described by $v_t = u_t G_0 + u_{t-1} G_1 + u_{t-2} G_2$. Any $(n_o, k_o, 2m)$ convolutional code with encoding matrices $g_0, g_1, \ldots, g_{2m}$ is equivalent to the $(mn_o, mk_o, 2)$ DMC with encoding matrices

$$G_0 = \begin{bmatrix} g_0 & g_1 & \cdots & g_{m-1} \\ 0 & g_0 & & g_{m-2} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & g_0 \end{bmatrix} \quad G_1 = \begin{bmatrix} g_m & g_{m+1} & \cdots & g_{2m-1} \\ g_{m-1} & g_m & & g_{2m-2} \\ \vdots & & \ddots & \vdots \\ g_1 & g_2 & \cdots & g_m \end{bmatrix}$$

$$G_2 = \begin{bmatrix} g_{2m} & 0 & \cdots & 0 \\ g_{2m-1} & g_{2m} & & 0 \\ \vdots & & \ddots & \vdots \\ g_{m+1} & \cdots & & g_{2m} \end{bmatrix}.$$

The state complexity of the DMC is $2mk_o$. The free distance is again given by the minimum weight vector resulting from an information sequence that is non-zero in the 0^{th} subblock. For an $(n, k, 2)$ DMC, because the set of such u_0's and their associated outputs can be considered as a $(3n, k)$ block code with $v_{0,2} = u_0[G_0 G_1 G_2]$, the optimal d_{free} is upper bounded by the highest attainable minimum distance of a $(3n, k)$ block code. The bounds are shown in Table 1.

The bound for $k = 1$ is uninteresting because an $(n, 1, 2)$ basic code is also an $(n, 1, 2)$ DMC. The bound is tight for most values of $k > 1$. However, in some cases, this block code upper bound for the DMC d_{free} is larger than the free distance achieved by the optimal basic code with the same rate and state complexity. In addition, the block code upper bound for DMCs is greater than or equal to the block code upper bound for UMCs, which indicates that larger free distances might be attained with DMCs. However, the existance of a DMC that attains the block code upper bound is not guaranteed. An exhaustive search for the double memory codes with maximal free distance is being conducted for small complexity values and results will be presented. In addition, a free distance bound for DMCs with uneven memory distributions is currently being studied.

References

[1] L.N. Lee, "Short Unit-Memory Byte-Oriented Binary Convolutional Codes Having Maximal Free Distance," *IEEE Trans. Inf. Theory*, May 1976, pp. 349-352.

[2] D.G. Mills and D.J. Costello.Jr., "Using a Modified Transfer Function to Calculate the Unequal Error Protection Capabilities of Convolutional Codes," submitted to the 1993 IEEE International Symposium on Information Theory.

[3] H.J. Helgert, and R.D. Stinaff, "Minimum-Distance Bounds for Binary Linear Codes," *IEEE Trans. Inf. Theory*, May, 1973, pp. 344-356.

[4] S. Lin and D.J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall,Inc., 1983.

R	n	k	K	DMC bound	optimal basic code [4]
1/2	2	1	2	6	5
	4	2	4	8	7
	6	3	6	10	10
	8	4	8	12	12
	10	5	10	15	14
	12	6	12	16	16
1/3	3	1	2	9	8
	6	2	4	12	12
	9	3	6	15	15
	12	4	8	18	18
	15	5	10	22	22
1/4	4	1	2	12	10
	8	2	4	16	16
	12	3	6	20	20
	16	4	8	24	24
2/3	3	2	4	6	5
	6	4	8	8	7
	9	6	12	12	--

Table 1

APPENDIX E

An Algebraic Approach to Constructing Convolutional Codes from Quasi-Cyclic Codes

An Algebraic Approach to Constructing Convolutional Codes from Quasi-Cyclic Codes [1]

Y. Levy D.J. Costello, Jr.
Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

June 10, 1992

Abstract — An algebraic approach to constructing convolutional codes from quasi-cyclic codes is investigated. Forney first discovered the interesting relationship between the structure of quasi-cyclic codes and convolutional codes. Recently, Tanner exploited this connection by comparing the syndrome matrices of both types of codes, which allows us to lower bound the free distance of a convolutional code with the minimum distance of an associated quasi-cyclic code. In this paper we first summarize Tanner's approach and show how it is possible to relate cyclic codes to convolutional codes by using quasi-cyclic codes as an intermediary. The problem of minimizing the constraint length of the convolutional codes is studied and an algorithm to find an equivalent convolutional code with reduced constraint length is presented. A number of codes with minimum constraint length and a lower bound on the free distance are found. However, there remains some open problems. For one, the actual free distance of some of the convolutional codes appears to be much higher than the lower bound indicates. Also, the rate of the convolutional codes is lower than the rate of the original quasi-cyclic codes and may lead to lower rate convolutional codes than expected.

Keywords — Convolutional Codes, Cyclic Codes, Quasi-Cyclic Codes.

[1]This work was supported by NSF Grant NCR89-03429 and by NASA Grant NAG5-557.

1. Introduction

Binary convolutional codes are employed in many communication systems because they provide efficient error correction and allow simple decoding algorithms. However, it is known that unlike block codes, for which numerous algebraic approaches lead to the construction of good codes, convolutional codes do not have a structure that allows a simple algebraic construction. In most applications, it is necessary to find good convolutional codes by exhaustive computer search, which limits the complexity of the codes obtained and their free distances.

Since no simple algebraic approach allows us to construct the generators of a convolutional code directly, one idea is to start from existing block codes and transform them into convolutional codes. Massey, Costello, and Justesen [1] used this approach in relating the generator of cyclic block codes to the n generators of rate 1/n convolutional codes. Then, Justesen [2] generalized this approach to the construction of q-ary rate l/n convolutional codes. However, in both of these approaches, the construction leads to good convolutional codes only for certain cyclic codes. The main problem in relating cyclic codes to convolutional codes is to construct from one cyclic code generator polynomial the ln generator polynomials of a rate l/n convolutional code. Thus, the idea of multiple generators for convolutional codes leads to the use of quasi-cyclic block codes, for which the structure is more similar to the structure of convolutional codes, thereby allowing us to use a larger subclass of block codes than the class of cyclic codes.

Tanner [3] extended the work of Forney [4] and Massey, Costello, and Justesen by relating the syndrome matrix of a convolutional code and a quasi-cyclic code. His main result was that the convolutional code constructed from the quasi-cylic code has a free distance lower bounded by the minimum distance of the quasi-cyclic code. The main problem with this approach, however, is finding the convolutional code with the minimum constraint length. In this paper, we investigate the different possibilities that exist in order to reduce the constraint length of the convolutional code, and we give an algorithm to convert a cyclic code into a convolutional code by using quasi-cyclic codes as an intermediary. Tables of the best codes constructed from BCH codes [5] are given, and the results are compared to optimal codes found by computer search [6][7].

2. Preliminaries

The notation introduced in this section is important to understand the connection between block and convolutional codes. We will first describe the syndrome matrix of quasi-cylic codes. Then, we will show how the syndrome matrix of a convolutional

code can be related to the syndrome matrix of a quasi-cyclic code in a particular form. For a comprehensive treatment of the theory of convolutional codes, the reader is referred to [6], and for quasi-cyclic codes, to [8].

2.1. Quasi-Cyclic Codes

Quasi-cyclic codes are a subclass of block codes which include cyclic codes [6] as a subclass. The defining property of quasi-cyclic codes is that one codeword can be obtained from any other codeword by cyclic shifts of n positions, which allows us to decompose a codeword into n blocks of m symbols, or m blocks of n symbols. Whether we choose the first or the second decomposition leads to a different form for the syndrome matrix. Note that if n=1, the quasi-cyclic code is cyclic.

A linear quasi-cyclic (N,K) code C_q, where $N = nm$ is the codeword length, n is the number of blocks in the codeword, and m is the length of each block, is the set of all n-tuples having the property defined above. That is, if

$$\mathbf{v} = (v_0, v_1, \ldots, v_{N-1}) \tag{1}$$

is a codeword in C_q, $\mathbf{v}$ shifted cyclically by n positions is also a codeword. In order to decompose the generator matrix into circulant blocks, we can reorder the indices of $\mathbf{v}$ to obtain

$$\mathbf{V} = (\underbrace{v_0, v_m, v_{2m}, \ldots, v_{(n-1)m}}_{1^{st}block}, \underbrace{v_1, v_{m+1}, \ldots, v_{(n-1)m+1}}_{2^{nd}block}, \ldots, \underbrace{v_{m-1}, v_{2m-1}, \ldots, v_{nm-1}}_{m^{th}block}). \tag{2}$$

Thus, a cyclic shift of $\mathbf{V}$ by n positions consists of moving the m^{th} block of n bits to the first block position, and the other blocks to the right by n positions. This yields

$$\mathbf{V} = (v_{in+j}), \tag{3}$$

for i=0 to n-1 and j=0 to m-1. A shift of n elements in $\mathbf{V}$ consists of adding 1 to j *mod* m. Such a shift takes any codeword to another codeword. Thus, it is also possible to write $\mathbf{V} = (v_{j+im})$ for j=0 to m-1 and i=0 to n-1. It follows from this notation that we can rearrange the vectors into the row space of a block matrix $\mathbf{G}$ consisting of kn circulant blocks $\mathbf{G}_j^{(i)}$ of size $m \times m$:

$$\mathbf{G} = \begin{bmatrix} G_0^{(0)} & G_0^{(1)} & \cdots & G_0^{(n-1)} \\ G_1^{(0)} & G_1^{(1)} & \cdots & G_1^{(n-1)} \\ \vdots & \vdots & \cdots & \vdots \\ G_{k-1}^{(0)} & G_{k-1}^{(1)} & \cdots & G_{k-1}^{(n-1)} \end{bmatrix}, \tag{4}$$

where $k = \left\lfloor \frac{K}{m} \right\rfloor$, $\lfloor\ \rfloor$ denotes the floor function, and for all i, $0 \leq i \leq n-1$, and j, $0 \leq j \leq k-1$, $G_j^{(i)}$ is an $m \times m$ circulant matrix, that is

$$G_j^{(i)} = \begin{bmatrix} g_{j0}^{(i)} & g_{j1}^{(i)} & \cdots & g_{jl}^{(i)} & 0 & \cdots & 0 \\ 0 & g_{j0}^{(i)} & g_{j1}^{(i)} & \cdots & g_{jl}^{(i)} & 0 & \\ & \ddots & \ddots & & & \ddots & \ddots \\ \ddots & & \ddots & \ddots & & & \ddots \\ & \ddots & & & \ddots & \ddots & \\ g_{j1}^{(i)} & \cdots & g_{jl}^{(i)} & 0 & \cdots & 0 & g_{j0}^{(i)} \end{bmatrix}. \tag{5}$$

The degree l polynomial

$$g_j^{(i)}(X) = g_{j0}^{(i)} + g_{j1}^{(i)}X + \ldots + g_{jl}^{(i)}X^l \tag{6}$$

is called the generator poynomial of the circulant matrix, with $l \leq m-1$.

The syndrome matrix $\mathbf{H}$ of the quasi-cylic code $\mathbf{C}_q$ is defined by $\mathbf{GH^T} = \mathbf{0}$ and is an $r \times n$ block matrix of $m \times m$ circulants, with $rm \geq N - K$. ($\mathbf{H}$ may contain linearly dependant rows.) In order to express the generator and syndrome matrices in a simpler manner, it is possible to use the polynomial form since the ring of circulant $m \times m$ matrices over the binary field is isomorphic to the ring of polynomials of degree l less than m, as noted in (6). Therefore, the syndrome matrix $\mathbf{H}(X)$ can be written as:

$$\mathbf{H}(X) = \begin{bmatrix} h_0^{(0)}(X) & h_0^{(1)}(X) & \cdots & h_0^{(n-1)}(X) \\ h_1^{(0)}(X) & h_1^{(1)}(X) & \cdots & h_1^{(n-1)}(X) \\ \vdots & \vdots & \cdots & \vdots \\ h_{r-1}^{(0)}(X) & h_{r-1}^{(1)}(X) & \cdots & h_{r-1}^{(n-1)}(X) \end{bmatrix}. \tag{7}$$

In the next section, we will see how the syndrome matrix of a convolutional code can be related to the syndrome matrix of the quasi-cylic code shown in (7).

2.2. Convolutional Codes

Let C_v be a rate l/n convolutional code. Then, for any information sequence $\mathbf{u}^{(i)}$, $1 \leq i \leq k$, that enters the encoder, the j^{th} encoded sequence is denoted $\mathbf{v}^{(j)}$, $1 \leq j \leq n$. By using polynomial notation, the sequence $\mathbf{u}^{(i)}$ can be written as

$$\mathbf{u}^{(i)}(X) = u_0^{(i)} + u_1^{(i)}X + \ldots + u_h^{(i)}X^h + \ldots, \tag{8}$$

and the encoded sequence $\mathbf{v}^{(j)}$ can be written as

$$\mathbf{v}^{(j)}(X) = v_0^{(j)} + v_1^{(j)}X + \ldots + v_h^{(j)}X^h + \ldots. \tag{9}$$

Thus, by denoting

$$\mathbf{u}(X) = (u^{(1)}(X), u^{(2)}(X), \ldots, u^{(l)}(X)), \tag{9}$$

and

$$\mathbf{v}(X) = (v^{(1)}(X), v^{(2)}(X), \ldots, v^{(n)}(X)), \tag{10}$$

we can write

$$\mathbf{u}(X) = \mathbf{G}(X)\mathbf{u}(X), \tag{11}$$

where $\mathbf{G}(X)$ is a $l \times n$ matrix of generator polynomials with the following form:

$$\mathbf{G}(X) = \begin{bmatrix} g_0^{(0)}(X) & g_0^{(1)}(X) & \cdots & g_0^{(n-1)}(X) \\ g_1^{(0)}(X) & g_1^{(1)}(X) & \cdots & g_1^{(n-1)}(X) \\ \vdots & \vdots & \cdots & \vdots \\ g_{l-1}^{(0)}(X) & g_{l-1}^{(1)}(X) & \cdots & g_{l-1}^{(n-1)}(X) \end{bmatrix}. \tag{12}$$

Like quasi-cyclic codes, the syndrome matrix $\mathbf{H}$ is defined by $\mathbf{G}(X)\,\mathbf{H}^T(X) = \mathbf{0}$, and has the following form:

$$\mathbf{H}(X) = \begin{bmatrix} h_0^{(0)}(X) & h_0^{(1)}(X) & \cdots & h_0^{(n-1)}(X) \\ h_1^{(0)}(X) & h_1^{(1)}(X) & \cdots & h_1^{(n-1)}(X) \\ \vdots & \vdots & \cdots & \vdots \\ h_{r-1}^{(0)}(X) & h_{r-1}^{(1)}(X) & \cdots & h_{r-1}^{(n-1)}(X) \end{bmatrix}, \tag{13}$$

where $r = n - l$.

Note that the structure of $\mathbf{H}(X)$ in (7) and (13) is identical. This suggests a connection between the two classes of codes. However, the number of rows r in both matrices is not defined the same way, which will affect the rate of the convolutional code constructed from a quasi-cyclic code, and the degree of each polynomial $h_j^{(i)}(X)$, for $0 \leq j \leq r-1$ and $0 \leq i \leq n-1$, is less than m in the quasi-cylic code syndrome matrix, whereas it can be of any degree in the convolutional code syndrome matrix.

3. Connection between Convolutional Codes, Quasi-Cyclic Codes, and Cyclic Codes

Tanner [3] proved that the similarity of structure between the syndrome matrices of quasi-cyclic and convolutional codes could be used to construct convolutional codes

from quasi-cyclic codes and vice versa. The goal of this section is to summarize his results and explain some of the remaining difficulties. We first make the connection between the two classes of codes and give Tanner's main theorems. Since the purpose of this paper is not to derive Tanner's results, we will simply state the main theorems without proof. We will study how the theory can be implemented to construct convolutional codes from cyclic codes by using quasi-cyclic codes as an intermediary, and finally, we will present an algorithm that minimizes the constraint length of the convolutional codes constructed.

3.1. The Syndrome Matrix Connection between Quasi-Cyclic Codes and Convolutional Codes.

The syndrome matrix is useful for block codes, especially to determine if a received vector belongs to the set of codewords defined by the code or if it needs to be corrected. In particular, a received vector $\mathbf{r}(X)$ belongs to a block code $\mathbf{C}_b$, whose syndrome matrix is $\mathbf{H}(X)$, if and only if

$$\mathbf{r}(X)\,\mathbf{H}^T(X) = \mathbf{0}. \tag{14}$$

As for block codes, a received sequence $\mathbf{r}(X)$ belongs to a convolutional code $\mathbf{C}_v$, whose syndrome matrix is $\mathbf{H}(X)$, if and only if it satisfies (14).

Tanner's approach to relating the syndrome matrices of quasi-cyclic codes and convolutional codes is to eliminate the problem that polynomials in a convolutional code syndrome matrix are of any finite degree by reducing these polynomials modulo $X^m + 1$, that is by taking the remainder of the division of each polynomial by $X^m + 1$. The first step is to construct a quasi-cylic code from a convolutional code by this modular reduction, which allows us to derive theorems relating to the minimum distance and the rate of the quasi-cyclic code. Then, by applying the reverse transformation from a quasi-cyclic code to a convolutional code, it is possible to find corollaries to these theorems which lower bound the free distance of the convolutional code.

Let $\mathbf{C}_v$ be a rate l/n convolutional code with syndrome matrix $\mathbf{H}_v(X)$ and $\mathbf{C}_q$ be the associated (N,K) quasi-cyclic code defined by the syndrome matrix $\mathbf{H}_q(X)$ constructed by reduction modulo $X^m + 1$ of $\mathbf{H}_v(X)$, where $m = N/n$ represents the size of the circulants of $\mathbf{H}_q$.

Theorem 1. *The rate l/n of the convolutional code $\mathbf{C}_v$ is less than or equal to the rate K/N of the quasi-cyclic code $\mathbf{C}_q$.*

Theorem 2. *The free distance of the convolutional code $\mathbf{C}_v$ is greater than or equal to the minimum distance of the quasi-cyclic code $\mathbf{C}_q$.*

Theorem 2 suggests that it is also possible to construct a convolutional code for which the modular reduction leads to a quasi-cyclic code with known minimum

distance, and therefore the free distance of the convolutional code is lower bounded by the minimum distance of the quasi-cyclic code. This leads to the following corollary:

Corollary 1. *Let $\mathbf{C}_q$ be a (nm,K) quasi-cyclic code with minimum distance d_{min} defined by an $r \times n$ syndrome matrix $\mathbf{H}_q(X)$ of polynomials of degree at most m-1. Let $\mathbf{C}_v$ be the convolutional code defined by the $r \times n$ syndrome matrix $\mathbf{H}_v(X)$ such that*

$$\mathbf{H}_v(X) = \mathbf{H}_q(X). \tag{15}$$

If $n-l$ is the rank of $\mathbf{H}_v(X)$, then C_v is a rate l/n convolutional code with $d_{free} \geq d_{min}$.

This corollary provides a very helpful method of constructing convolutional codes from quasi-cylic codes. However, the problem of the constraint length ν has not been studied yet. In fact, the only information on the constraint length of the resulting convolutional codes come from the size of the quasi-cyclic code circulants m. Indeed,

$$\nu \leq l\,(m-1), \tag{16}$$

since the maximum degree of the polynomials of $\mathbf{H}_q(X)$ is m-1, and the number of information sequences is l.

In order to minimize the constraint length of the resulting convolutional codes, Tanner [3] suggested constructing quasi-cyclic codes for the purpose of finding convolutional codes. Indeed, numerous lists of quasi-cyclic codes have already been published, and they could easily be used in the construction described by Corollary 1. However, Tanner also showed that it is possible to convert block cyclic codes into quasi-cyclic codes and thereby find the quasi-cyclic code realization that gives the convolutional code with the highest rate and the smallest constraint length. We now briefly summarize these results of Tanner, which then lead to our construction algorithm.

3.2. Construction of quasi-cyclic codes from cyclic codes

Starting from a cyclic code of composite length $N = nm$, it is possible to construct a quasi-cyclic code as described in the following lemma.

Lemma 1. *Any $nm \times nm$ circulant matrix is equivalent under row and column permutations to an $n \times n$ block matrix of $m \times m$ circulants.*

The corollary of this lemma allows us to transform the syndrome matrix of a cyclic code into the syndrome matrix of a quasi-cyclic code.

Corollary 2. *Any rate K/nm cyclic code is equivalent to a quasi- cyclic code defined by an $n \times n$ syndrome matrix of $m \times m$ circulants.*

Lemma 1 and Corollary 2 are important because they provide a way of transforming a cyclic code syndrome matrix into a quasi-cyclic code syndrome matrix. Indeed,

the actual syndrome matrix of a cyclic code of rate K/nm is of size $(nm-K)\times nm$. By extending the circulation of the parity-check polynomial, it is possible to obtain an $nm\times nm$ syndrome matrix $\mathbf{H}_c$ of rank $nm-K$. Then, each $m\times m$ block $\mathbf{H}_{q(I,J)}$ of the $nm\times nm$ quasi-cyclic code syndrome matrix $\mathbf{H}_q$, for $0\leq I,J\leq n-1$, is obtained by letting $\mathbf{H}_{q(I,J)}(i,j)=\mathbf{H}_c(I+in,J+jn)$ for $0\leq i,j\leq m-1$. In order to obtain the syndrome matrix of a rate $(nm-rm)/nm$ quasi-cyclic code, it is necessary to remove $l=n-r$ row blocks of m rows each from the matrix $\mathbf{H}_q$, while keeping the rank equal to $nm-K$. This means that $l\leq\left\lfloor\frac{K}{m}\right\rfloor$. (See Example 1.)

This last operation consisting of deleting blocks of rows is the most tedious, since it is not obvious which blocks can be deleted without decreasing the rank, or which blocks one should delete in order to construct the convolutional code with the smallest constraint length. One way of checking the rank of the matrix is to use Tanner's transform theory developed in [9] which involves Galois Field Algebra that is not simple to implement. Another way is to transform $\mathbf{H}_q$ into systematic form

$$\mathbf{H}_q^s=\begin{bmatrix}\mathbf{I}_s:\mathbf{A}\\ \cdots\cdots\\ \mathbf{D}\end{bmatrix} \qquad (17),$$

where s is the rank of $\mathbf{H}_q^s$, $\mathbf{I}_s$ is the identity matrix of size $s\times s$, $\mathbf{A}$ is of dimension $s\times(n-s)$, and $\mathbf{D}$ is of dimension $(n-s)\times n$. Linearly dependent rows of $\mathbf{H}_q$ can only appear in the last rows of the matrix, denoted by $\mathbf{D}$.

Once the syndrome matrix of the quasi-cyclic code is constructed, it is also possible to permute rows and columns within blocks of circulants without changing the distance or the rate of the quasi-cyclic code. This operation is particularly useful for the purpose of constructing a convolutional code with low constraint length, since the degree of the polynomials in $\mathbf{H}_q(X)$ must be as low as possible in order to construct a convolutional code with small constraint length.

Therefore, in the process of constructing a convolutional code from a cyclic code using a quasi-cyclic code as an intermediary, it is necessary to combine all these operations in a single algorithm. We present an algorithm for doing this in the following section.

4. An Algorithm To Construct Convolutional Codes from Cyclic Codes

In this section, we study how to combine the operations discussed previously to construct the convolutional code with the smallest constraint length starting from a given cyclic code.

4.1. Algorithm

As seen previously, once permutations have been performed on the rows and columns of the cyclic code syndrome matrix to construct the square syndrome matrix of the quasi-cyclic code (see steps 2 and 3), it is necessary to remove some row blocks of the quasi-cyclic code syndrome matrix; otherwise, the algorithm would lead to a rate zero convolutional code, since the generator matrix would have no rows. In order to remove the rows that correspond to the highest degree polynomials (since polynomials with high degrees lead to large constraint length convolutional codes), it is necessary to make permutations within blocks of circulants to reduce the degree of the polynomials as much as possible (see steps 4,5,6, and 7). Then, we try to remove the blocks with the highest degree polynomials (see steps 8,9, and 10). Finally, if the rank of the original cyclic code is maintained, a new permutation within blocks of circulants might decrease the degrees of the polynomials again (see step 11). This leads to the following algorithm.

Algorithm:

Step 1: Select an (nm.K) cyclic code with parity-check polynomial $\mathbf{h}(X)$ and minimum distance d_{min}.

Step 2: Construct the $nm \times nm$ matrix $\mathbf{H}_c$ by putting $\mathbf{h}(X)$ in the first row and its successive cyclic shifts in the remaining rows.

Step 3: Construct the $n \times n$ block matrix $\mathbf{H}_q$ of $m \times m$ circulants using Corollary 2.

Step 4: Convert $\mathbf{H}_q$ to its polynomial form $\mathbf{H}_q(X)$.

Step 5: Construct the vectors $\mathbf{R} = (r_0, \ldots, r_{n-1})$ and $\mathbf{C} = (c_0, \ldots, c_{n-1})$, where $r_I = \max_{0 \le J \le n-1} \deg \mathbf{H}_{q(I,J)}(X)$, and $c_J = \max_{0 \le I \le n-1} \deg \mathbf{H}_{q(I,J)}(X)$, $0 \le I, J \le n-1$. Let $r = \max_{0 \le I \le n-1} r_I$ and $c = \max_{0 \le J \le n-1} c_J$.

Step 6: Permute rows within block I of circulants of $\mathbf{H}_q$, for $0 \le I \le n-1$, until r is minimal over all possible permutations.

Step 7: Repeat Step 6 with columns and minimize c.

Step 8: Let $l = \left\lfloor \frac{K}{m} \right\rfloor$.

Step 9: Delete l row blocks I from $\mathbf{H}_q$, where I is the index of the largest l values of r_I.

Step 10: Compute the rank of $\mathbf{H}_q$. If the rank is $nm - K$, go to Step 11. If the rank is strictly less than $nm - K$, go back to Step 9 and try another set of row blocks. If all sets of l row blocks have been tried, set $l = l - 1$ and go back to Step 9. If $l = 0$, then the construction is impossible.

Step 11: $\mathbf{H}_q$ is an $(n - l) \times n$ block matrix of $m \times m$ ciculants. Repeat Steps 4,5,6, and 7 on $\mathbf{H}_q$ with $0 \leq I \leq n - 1 - l$ instead of n.

Step 12: Let $\mathbf{H}_v(X)$=$\mathbf{H}_q(X)$ be the syndrome matrix of a rate l/n convolutional code with $d_{free} \geq d_{min}$.

Step 13: Construct a generator matrix $\mathbf{G}_v(X)$ such that $\mathbf{G}_v(X)\mathbf{H}_v^T(X) = \mathbf{0}$.

Step 14: Convert $\mathbf{G}_v(X)$ to minimal form (see [10] and [11]). Stop.

Note that Steps 6 and 7 can be exchanged without modification of the result. Note also that step 11 consists in permuting rows and columns like in steps 4,5,6, and 7, except that the number of rows in $\mathbf{H}_q$ is now only $n - l$.

4.2. Example

In order to fully understand this algorithm, we now give an example of the construction of a convolutional code, starting from the (15,5) BCH code with minimum distance $d_{min} = 7$, originally taken as the example in Tanner's paper [3].

Example 1. *Let C_c be the (15,5) BCH code.*

Step 1: $\mathrm{h}(X) = 1 + X + X^3 + X^5$, that is $\mathbf{h}$=[110101], $n = 3$, $m = 5$, and $K = 5$.

Step 2: $\mathbf{H}_c =$

$$\begin{bmatrix}
1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}.$$

Step 3: $\mathbf{H}_q =$

$$\left[\begin{array}{ccccc|ccccc|ccccc}
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
\hline
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{array}\right].$$

Step 4: $\mathbf{H}_q(X) = \begin{bmatrix} 1+X^4 & 1 & X^4 \\ X^3 & 1+X^4 & 1 \\ X^4 & X^3 & 1+X^4 \end{bmatrix}.$

(Note that in $\mathbf{H}_q$, the polynomials correspond to the first column of each circulant.)

Step 5: $\mathbf{R} = (4, 4, 4)$ and $\mathbf{C} = (4, 4, 4)$, so $r = 4$ and $c = 4$.

Step 6 and 7: $\mathbf{H}_q =$

$$\left[\begin{array}{ccccc|ccccc|ccccc}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0
\end{array}\right].$$

$\mathbf{R} = (1, 2, 2)$ and $\mathbf{C} = (1, 2, 2)$.

Step 8: $l = 1$.

Step 9: $\mathbf{H}_q =$

$$\left[\begin{array}{ccccc|ccccc|ccccc}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0
\end{array}\right].$$

Step 10: $rank = 10 = nm - K$.

Step 11: No modification from Step 9. $\mathbf{H}_q$ stays the same.

Step 12: $\mathbf{H}_v(X) = \mathbf{H}_q(X) = \begin{bmatrix} 1+X & X & 1 \\ X & 1 & X+X^2 \end{bmatrix}$.

Step 13: $\mathbf{G}_v(X) = \begin{bmatrix} 1+X+X^2 & 1+X^2+X^3 & X^3 \end{bmatrix}$.

Step 14: $\mathbf{G}_v(X)$ is already in minimal form. The constraint length of the rate 1/3 convolutional code is 3, and $d_{free} \geq 7$. Actually, in this particular case, $d_{free} = 7$.

This example results in a constraint length of only 3, whereas the upper bound is 4 from (16). The free distance equals its lower bound, which is not the case in general, but the best rate 1/3 convolutional code with constraint length 3 has a free distance of 10 [6]. The algorithm was also applied to BCH codes of length 15, 63, and 255, since their lengths are composite. Specifically, $15 = 3 \times 5$, $63 = 3 \times 3 \times 7$, and $255 = 5 \times 3 \times 17$, which allows us to construct codes with rates k/3, k/5, k/7, and k/9. The other rates that can be constructed from these codes, such as k/15, k/17, or k/21, are very low and thus were not considered. In the next section, we give tables of convolutional codes constructed from these classes of BCH codes.

5. Results and Comments on Construction of Convolutional Codes from BCH Codes

The tables of constructed codes show the code rate, the generator matrix with polynomials in octal form (*e.g.*, 54 corresponds to 101100, *i.e.*, $1+X^2+X^3$), the constraint length, the memory order, the original BCH code, the lower bound on the free distance, the computed free distance (for some codes [2]), and the best code found by exhaustive search. In this last column, either the free distance of the best code with the same rate and constraint length is given or the constraint length ν of the best code with the same rate and free distance is given.

[2] X means unknown

5.1. Construction Table for Galois Field GF(16)

Rate	Generator Matrix	Const. Length	Mem. Order	BCH Code	Lower Bd on d_{free}	Comp. d_{free}	Best Code
1/3	54 1 7	3	3	(15,5)	7	7	10 [6]
1/5	1 4 7 54 64	3	3	(15,5)	7	11	16 [7]
1/5	4 2 2 7 4	2	2	(15,5)	7	7	13 [7]
2/5	4 0 4 0 6 0 0 4 4 4	1	1	(15,11)	3	3	4 [7]
2/3	0 6 4 6 4 0	2	1	(15,11)	3	3	3 [6]
3/5	4 6 2 0 0 4 4 0 2 0 6 4 0 0 2	2	1	(15,11)	3	3	4 [7]

5.2. Construction Table for Galois Field GF(64)

Rate	Generator Matrix	Con. Len.	Mem. Ord.	BCH Code	Bd d_{fr}	Com. d_{fr}	Best Code
2/9	11 34 63 12 46 56 44 62 14 34 63 12 46 56 44 62 14 22	10	5	(63,18)	21	X	X
1/9	11 34 63 12 46 56 44 62 14	5	5	(63,18)	21	25	X
2/9	2 12 3 4 44 6 62 14 6 12 3 4 44 6 62 14 11 4	8	4	(63,24)	15	X	X
3/9	3 4 44 6 62 14 6 4 24 12 3 4 44 6 62 14 6 4 2 12 3 4 44 6 62 14 6	12	4	(63,24)	15	X	X
1/7	44 0 66 76 52 74 62	4	4	(63,30)	13	21	28 [7]
2/7	44 0 66 76 52 74 62 26 44 66 1 24 26 54	8	4	(63,30)	13	X	ν=4 [7]
3/7	44 0 66 76 52 74 62 72 62 22 76 34 02 47 26 44 66 1 24 26 54	13	5	(63,30)	13	X	X
4/7	02 3 1 32 64 56 6 51 4 04 04 4 43 1 36 26 7 74 62 72 14 74 64 4 34 0 6 04	16	5	(63,36)	11	11	X
6/7	4 0 0 0 0 2 4 4 0 2 0 0 4 0 6 4 2 0 0 2 0 0 4 4 0 0 4 4 6 2 0 4 0 0 0 4 0 4 0 4 4 0	4	1	(63,57)	3	3	X
5/7	0 4 0 0 0 0 6 4 0 0 0 0 4 4 0 2 0 4 0 4 6 0 0 0 4 4 0 4 0 2 4 4 0 2 0	3	1	(63,57)	3	3	4 [7]
2/3	4 4 2 0 6 4	2	1	(63,57)	3	3	3 [6]

5.3. Construction Table for Galois Field GF(256)

Rate	Generator Matrix	Con. Len.	Mem. Ord.	BCH Code	Bd d_{free}	Comp. d_{free}	Best Code
2/3	6123 637 0442 0665 64614 76646	24	13	(255,215)	11	X	X
1/3	62564 76204 3754	12	12	(255,215)	11	20	24 [6]
2/3	7632 314 735 2004 465 4415	21	11	(255,223)	9	X	ν=9 [7]
3/5	14 7 44 34 36 07 674 6 534 4 3 3 76 7 1	14	6	(255,231)	7	X	ν=5 [7]
1/3	641 716 354	8	8	(255,231)	7	14	18 [6]
2/3	554 7364 152 461 3 46	17	9	(255,231)	7	X	ν=6 [6]
1/3	71 15 74	5	5	(255,239)	5	10	13 [6]
2/5	77 614 454 0 602 601 073 066 602 0	15	8	(255,239)	5	X	X
1/5	247 216 362 662 54	8	8	(255,239)	5	22	ν=5 [7]
2/3	72 75 26 6 32 56	9	5	(255,239)	5	X	9 [6]
3/5	0 26 26 64 54 34 6 44 1 0 6 64 2 4 5	10	4	(255,239)	5	X	ν=3 [7]
4/5	22 0 42 24 76 2 6 0 1 6 4 7 7 5 3 54 0 1 34 6	11	4	(255,239)	5	X	X

5.4. Comments on the Constructed Codes

Some conclusions can be drawn from the tables above. On the one hand, some of the codes cannot be compared to codes found by exhaustive search, since they have a larger constraint length than any code of the same rate found by exhaustive search. Thus, this algebraic construction allows us to construct codes that are impossible to find by exhaustive search. On the other hand, the codes for which a comparable optimal code exists usually have a suboptimal ratio of free distance to constraint length, although for some constructed codes only the lower bound on the free distance

is known, which may be weak compared to the actual free distance of the code. For example, the rate 1/5 code with constraint length 8 has free distance 22, whereas the bound only gives 5.

It was also observed, especially for GF(256), that only very high rate BCH codes lead to the construction of quasi-cyclic codes and convolutional codes. This means that it is hard to find convolutional codes of low rate with very high free distance, since high rate block codes have low minimum distance. More generally, it was observed that a large number of BCH codes do not lead to the construction of quasi-cyclic codes with lower cycle lengths m, and it is necessary to use BCH codes with much higher rates than the convolutional codes constructed. For example, the first 2/3 convolutional code constructed from GF(256) was based on a (255,215) code, whereas it should theoretically be possible to start with the (255,171) BCH code, which would lead to a higher free distance.

Finally, it was observed that this construction leads to good codes for rates close to 1/2. In particular, the rate 3/7 and 4/7 codes constructed from the (63,30) and (63,36) BCH codes have a large constraint length and a good, although suboptimal, ratio of free distance to constraint length.

6. Conclusions

We have proposed a construction algorithm for convolutional codes based on Tanner's discovery of the connection between cyclic codes, quasi-cyclic codes and convolutional codes. Codes have been constructed with this algorithm, and some conclusions have been drawn about this new algebraic approach to constructing convolutional codes from block codes. In particular, this algebraic construction allows us to construct non-standard rate convolutional codes with large constraint length, but in general leads to suboptimal codes.

One might wonder if trying to relate quasi-cyclic codes to convolutional codes does not lead to very good convolutional codes because the quasi-cyclic codes used are not designed for the construction of convolutional codes. Thus, the problem of designing good quasi-cyclic codes with the purpose of constructing good convolutional codes should be investigated.

7. Acknowledgements

The authors want to acknowledge the work of Michael Tanner who first derived the main theorems of this paper.

References

[1] J.L. Massey, D.J. Costello, Jr., and J. Justesen, "Polynomial Weights and Code Constructions," *IEEE Trans. Inf. Theory*, IT-19, pp.101-110, Jan. 1973.

[2] J. Justesen, "New Convolutional Code Constructions and a Class of Asymptotically Good Time-Varying Codes," *IEEE Trans. Inf. Theory*, IT-19 Number 2, pp. 220-223, March 1973.

[3] R.M. Tanner, "Convolutional Codes from Quasi-Cyclic Codes: A Link Between the Theories of Block and Convolutional Codes," *Computer Research Laboratory, Technical Report, UCSC-CRL-87-21*, November 1987.

[4] G.D. Forney, Jr., "Why Quasi-Cyclic Codes are interesting," privately circulated unpublished note (circa 1970).

[5] R.C. Bose and D.K. Ray-Chaudhuri, "On a Class of Error Correcting Binary Group Codes," *Inf. Control*, 3, pp.68-79, March 1960.

[6] S. Lin, and D.J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, pp.322-327, Prentice-Hall, 1983.

[7] D.G. Daut, J.W. Modestino, and L.D. Wismer, "New Short Constraint Length Convolutional Code Constructions for Selected Rational Rates," *IEEE Trans. Inf. Theory*, IT-28, pp.794-800, Sept. 1982.

[8] C.L. Chen, W.W. Peterson, and E.J. Weldon, Jr., "Some Results on Quasi-Cyclic Codes," *Inform. Control* , 15, pp.407-423, November 1969.

[9] R.M. Tanner, "A Transform Theory for Group-Invariant Codes," *IEEE Trans. Inf. Theory*, IT-34, No. 4, pp.752-775, July 1988.

[10] R. Johannesson, and Z.X. Wan, "A Linear Algebra Approach to Minimal Convolutional Encoders," *IEEE Trans. Inf. Theory*, to be published.

[11] G.D. Forney, Jr., "Convolutional Codes I: Algebraic Structure." *IEEE Trans. Inf. Theory*, IT-16, pp.720-738, November 1970.